

# Natural Language Processing

## **Semantic search engine based on Wikipedia (selected topic).**

T. S. & E. B.

# Introduction

## **1 - Abstract – a short description of the work**

Semantic analysis is a real challenge. It involves finding the meaning of a sentence, rather than simply analyzing the sentence to find the most numerous occurrences.

You have to work on the sentence to understand it as well as possible, comparing it with other texts to best meet the user's requirements.

## **2 - Aim– describe the purpose of the project**

The aim of this project is to analyze a sentence using semantic analysis. To do this, the user will search for something (ask a question) and the program will search a database of Wikipedia pages on the countries of the world and try to find the country closest to the query.

## **2 Scope – describe the scope that you covered in the project**

A search using semantic analysis can have extremely varied applications. It can, for example, be the start of a chat, help to get more precise answers in a search (as we're doing here), or, in a search browser, enable you to respond directly to the user's expectations without returning sites. For example, if you google “Barack Obama's daughter”, you get their name directly, rather than pages containing “daughter” and “Obama”.

## **3 Methodology – methodology and tools we’ve used**

To perform a semantic analysis successfully, we need to go through several steps.

First, of course, we need to read the data we've downloaded. We've created a program to retrieve text from various Wikipedia pages.

Once we've retrieved all this data, we also need to process it. We delete all the stop words and put the text in lower case. This prevents the same word from being listed several times.

We then build a TF-IDF vector with all the words, to make semantic analysis much easier later, and to enable better exploitation of the data.

We proceed in the same way with the user input. We recover all the words, deleting the Word stops.

Once we've obtained two sets of data that can be processed by a computer, we then need to compare them to find out which subject comes closest to the user input. To do this, we use the “cosine\_similarity” function. This function automatically takes two TF-IDF vectors and compares them to find which subject is closest to the 2nd. Once this text is found, we simply retrieve from the database which url this comparison corresponds to and return the requested query to the user.

Then, we can apply semantic analysis. We are using the PCA method. It provides a better result than a simple LSA.

The PCA method create several topics, with the TF-IDF vector, which contains many different words. After that, we associate this with the query, and associate a value depending on the topics. The bigger value is chosen, and then the topic associated with the value is chosen, which can return a country.

We decided not to implement a graphical interface. This is not the aim of the project and would simply complicate the program's task. Development would have taken longer, with a less qualitative result.

## **Theoretical Part**

### **4 - the theory we used to solve the problem.**

Our semantic analysis works quite well. On rather easy topics, such as political leaders or capital cities, it works perfectly well. The same goes for well-documented major events. The program is able to link the user's input to the desired URL.

Like any computer program, our code obviously has its limits. Indeed, if the user isn't precise enough, or if he enters rather common words, our program returns erroneous results. What's more, although we don't know exactly why, it also returns random countries for certain cities.

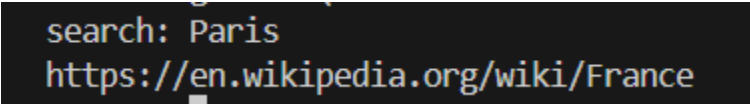
Below, you'll find screenshots of various examples of our program. Both examples that work well, and examples that return incorrect results.

## **Practical part**

### **5 - the solution to the problem, present the results**

The goal has been partially achieved. As you can see below, on fairly basic cases, the program returns the correct Wikipedia page. However, on slightly more complex questions, including names of senior executives, the program will not return the correct values.

Correct results :



```
search: Paris
https://en.wikipedia.org/wiki/France
```

```
search: Poland
https://en.wikipedia.org/wiki/Poland
```

```
search: Where is Paris
https://en.wikipedia.org/wiki/France
```

```
search: Which country speak Polish
https://en.wikipedia.org/wiki/Poland
```

```
search: Which country was invaded during WW2
https://en.wikipedia.org/wiki/Luxembourg
```

```
search: Which country was invaded during WWII and speak Polish
https://en.wikipedia.org/wiki/Poland
```

Incorrect results :

```
search: Which country has Emmanuel Macron as President ?
https://en.wikipedia.org/wiki/Ireland
```

(the good answer was France)

In addition, we have also made another program, which don't use PCA. And this program is more performant on some question, like the personalities or the places, but not on some other questions more complicated.

True answers :

```
Enter your query: Paris
The URL of the closest summary is: https://en.wikipedia.org/wiki/France
```

```
Enter your query: Poland
The URL of the closest summary is: https://en.wikipedia.org/wiki/Poland
```

```
Enter your query: Where is Paris ?
The URL of the closest summary is: https://en.wikipedia.org/wiki/France
Enter your query: Which country speaks Polish ?
The URL of the closest summary is: https://en.wikipedia.org/wiki/Poland
```

```
Enter your query: Which country has Macron as President ?
The URL of the closest summary is: https://en.wikipedia.org/wiki/France
```

```
Enter your query: Where is Notre-Dame ?
The URL of the closest summary is: https://en.wikipedia.org/wiki/France
```

Bad answers :

```
Enter your query: Where is Krakow ?
The URL of the closest summary is: URL
```

```
Enter your query: Which country was invaded during the second world war ?
The URL of the closest summary is: https://en.wikipedia.org/wiki/Spain
```

## 6 – Summary

I think we can safely say that we've achieved our goal. We've succeeded in making a search engine that takes a user query as input. The program manages to understand the meaning of what the user is asking for and then returns the link to the corresponding Wikipedia page. So we can make sentences without unnecessary or very common words getting in the way of the query. You can refer to our examples or test the program on simple questions. Of course, without competing with Google or other tech giants, we've still managed to meet the required specifications by offering a search engine that takes phrases as input and returns what the user wanted to express with that phrase.

However, if we try to push the limits of the program, we can find obvious cases where even seemingly simple queries can lead to erroneous results. We noticed, for example, that it didn't know how to handle cities very well. When searching for a city, it frequently misplaces it on the map.

## 7. Bibliography

- The course of NLP
- <https://www.geeksforgeeks.org/web-scraping-from-wikipedia-using-python-a-complete-guide/>
- <https://www.geeksforgeeks.org/implementing-pca-in-python-with-scikit-learn/>
- <https://pub.towardsai.net/understanding-semantic-analysis-using-python-nlp-f48016422677>